Hybrid Thinking in Vision-Language-Action Models

Anonymous Author(s)
Affiliation
Address
email

Abstract: Using Large Language Models to produce intermediate thoughts before providing a final answer has been a successful recipe for solving increasingly complex tasks with reduced human supervision. In robotics, similar embodied reasoning strategies have also been shown to lead to improved performance. However, as these techniques increase the length of the model's outputs to include reasoning traces, the inference time is negatively affected. Delaying an agent's actions in real-world executions, as in robotic manipulation settings, can be particularly problematic, as the agent needs to perform long sequences of actions before solving a task. In this work, we establish a Hybrid Thinking (HyT) framework for training Vision-Language-Action (VLA) models. Agents can learn both to directly answer with actions (fast mode) or to spend more time thinking (slow mode). We show that, even when generating no thoughts, in fast mode, the agent performance benefits from training on the reasonings that leads to successful actions. Our agent demonstrates improved performance at lower inference costs, and greater scalability with larger datasets across a set of different robotic manipulation tasks. Additionally, hybrid thinking allows humans to interpret the agents' intentions and intervene on them to prevent failures for complex tasks' execution.

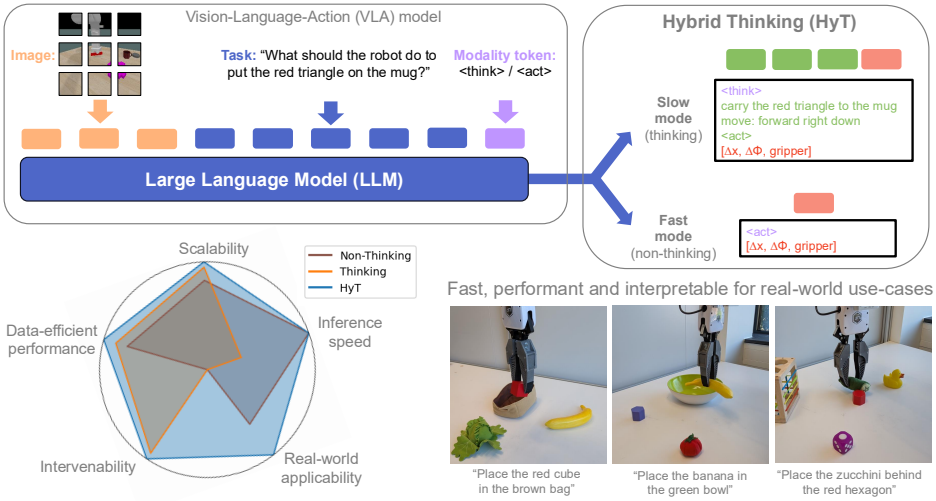


Figure 1: **Hybrid Thinking VLAs.** The hybrid thinking framework enables vision-language-action models to work either in fast, acting, mode or thoughtful, slow, execution. In fast mode, the agent retains high performance, at high inference speed, facilitating deployment in real-world platforms. In slow mode, the model grants high interpretability and the possibility of intervening on its thoughts, to change the predicted course of actions. Further details about the Figure are provided in the Appendix.

1 Introduction

Despite recent advances in robotics, truly generalist robot policies have long been elusive. Thanks to the joint efforts of collecting large-scale robot data [1] and making large Vision Language Models

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.

(VLM) open-source [2, 3], we have entered a new era in robotics foundation models. By fine-tuning VLMs on robotic datasets containing actions, we can create Vision-Language-Action models (VLAs) [4, 5, 6]. These large policy models are trained end-to-end to take language instructions and raw camera images as inputs, and output low-level actions for the robot to perform.

VLAs possess several advantages over previous work, such as multimodal prompting of the agent and the availability of knowledge from the base pre-trained VLM. However, generalization to out-of-distribution (OOD) settings, e.g., task configurations not available in the robotics training dataset, remains challenging. Indeed, the knowledge of the agent is vast about general concepts, but remains limited in the robotics settings, where the data distribution is often narrow.

In order to further unleash the capabilities of VLAs, recent works have explored adding Chain-of-Thought (CoT) reasoning [7] while training VLAs [8]. This class of *thinking VLA* models learns to output useful information about the given task in language form, before generating the actions to execute. This not only has shown to improve performance, but it also allows humans to more easily interpret the agent's intentions and potentially intervene on them, i.e. manipulating the agent's thoughts, before action generation. However, due to the large amount of reasoning outputs generated before actions, the inference time of these models is significantly higher.

Similarly to thinking VLAs, *hierarchical VLA* methods [9, 10] aim to improve performance by leveraging a two-level system. A high-level VLM processes the instructions and the information from the environment and provides an actionable plan. A low-level VLA policy receives the higher-level plan as an instruction and generates robotic actions to execute accordingly. This class of VLAs offers similar benefits and drawbacks as thinking VLAs: they can improve performance and allow humans to read and/or manipulate the agent's intentions, but they come with an even higher inference cost.

The human cognition process from observation to action has been hypothesized to leverage the interaction of two systems [11]. The fast and intuitive *System I* handles most daily tasks, taking control in contexts that our brain judges as unchallenging. The slow and deliberate *System II* is activated when decisions require additional computation, such as comparing options or processing complex information. The tendency of the brain is to delegate decisions to System II only when it's really necessary, to save energy and time. Humans can improve the capabilities of their System I, developing a skilled intuition [12] to solve complex but familiar tasks effortlessly.

We hypothesize that VLA models can similarly develop more skilled intuition. Learning from the CoT reasoning traces, a model should be able to internalize knowledge about environments and tasks. Then, during test-time, the model should more easily recognize patterns, even without generating any thoughts. With this hypothesis in mind, we develop a **hybrid thinking** (HyT) framework, where the agent learns to operate in different modes, thinking and non-thinking ones, within a single model.

Hybrid thinking can be effectively implemented by teaching the model to predict a variety of outputs, which are sampled with different probabilities during training. During test-time, the model can operate in different modes: primarily a fast mode and a slow mode. The fast mode improves performance over standard VLAs, while having no higher inference costs. The slow mode could be employed in critical settings, where interpreting the agent's intentions or allowing interventions from humans could be useful. In addition to investigating the hybrid thinking framework, we aim to address a fundamental question regarding VLA models: *What is the contribution of reasoning and CoT techniques to their performance?*

Our contributions are:

- We establish the hybrid thinking (HyT) framework for vision-language-action (VLAs) models. We present an implementation of HyT which has a Fast and a Slow mode, enabling both fast action inference and rational thinking when necessary, within the same model.
- We empirically validate our approach, highlighting its data-efficiency, scalability and intervenability across a large quantity of simulated experiments.
- We perform a real-world experiment on a UFactory xArm 6 that showcases the applicability of the approach in real robotics settings.

2 Related Work

Vision-Language-Action models. Open-source efforts in the robotics field, such as the Open X-Embodiment dataset [1], have fueled progress in the development of large VLAs [4, 13, 14, 15, 16]. Recent works have also explored hierarchical VLA architectures [9, 10], showing they can be beneficial for solving open-ended and long-horizon tasks.

Chain-of-Thought and reasoning. Generating a chain of thought has shown improved performance in LLMs solving complex reasoning tasks [17]. Recently, reasoning has shown notable success using RL with verifiable rewards, coupled with Supervised Finetuning (SFT) on example reasoning traces [18, 19]. CoT techniques specifically for VLMs [20] and VLAs have also been researched [8, 21]. In particular, ECoT [8] shows that embodied thoughts can greatly improve the agent's predictions in robotics, despite the higher inference costs. Our work grounds on their findings and proposes a method that accomplishes both strong performance and fast inference.

Hybrid reasoning. Recent works have attempted to distill slow thinking capabilities into faster models [22, 23]. Closely related to our method is DualFormer [24], proposes to train a language model by systematically dropping out reasoning traces. In robotics domain, RFST [25] proposes a hierarchical setup that uses a discriminator to decide whether to switch to the fast or slow system, with the respective model of the chosen mode being then used as the policy. Our work, instead, focuses on providing a single system that is capable of both thinking and acting.

3 Method

3.1 Formal definitions

Vision-Language Action models are multimodal policies generally trained with imitation learning. A VLA processes language inputs through a Transformer-based LLM architecture [26, 27, 28]. Language is first "tokenized" into language tokens that are then processed by the LLM. Similarly, VLAs can process visual inputs through a vision encoder, e.g., a vision transformer [29], that transforms image patches into visual tokens, which are then processed by the LLM.

Given a language description of a task ℓ, the goal of the VLA policy is to solve the task in a given environment. The policy observes the environment through images x, generally captured by a camera in the environment. The policy interacts with the environment using actions a. Through imitation learning, the policy's objective is to learn, at each discrete timestep, the distribution $p(a_t|x_t, \ell)$ that solves the given task, which is empirically observed from a dataset of demonstrations. In this work, we assume that actions are mapped to tokens in the LLM's vocabulary, through a discretization scheme that assigns continuous values to one of the 256 bins [4]. This enables the LLM to predict action tokens in the same vocabulary space as language tokens. A VLA policy can be defined as:

$$\text{VLA:} \qquad p(a_t|x_t, \ell) = p_\theta(a_t|x_t, \ell).$$

In addition to predicting actions, thinking VLAs [8] are also capable to reason about the task and the environment. These reasonings are expressed as thoughts τ in language form, predicted by the model. Generally, thoughts include information about the overall plan of action, the current subtask to execute, the location of objects in the image, or the direction of the agent's ongoing motion [8]. A thinking VLA policy can be defined as:

$$\text{Thinking VLA:} \qquad p(a_t, \tau_t|x_t, \ell) = p_\theta(a_t|x_t, \ell, \tau_t)p_\theta(\tau_t|x_t, \ell).$$

Thinking VLAs learn a single set of parameters θ to predict both actions and thoughts. Hierarchical VLAs [9, 10] use a two-level hierarchy of models, where one model is to provide an actionable plan for solving the task, while the second model should execute the plan. As shown in [9], predicting the current plan to solve the task can be as simple as predicting the current subtask and motion primitives. Thus, we treat high-level plans and thoughts interchangeably in this work. A hierarchical VLA policy can be defined as:

$$\text{Hierarchical VLA:} \qquad p(a_t, \tau_t|x_t, \ell) = p_{\theta_\ell}(a_t|x_t, \tau_t)p_{\theta_h}(\tau_t|x_t, \ell),$$

where θ_h denotes the parameters of the "high-level" model, θ_ℓ denotes the parameters of the "low-level" model, and the models' hierarchy enforces $(a_t \perp\!\!\!\perp \ell | \tau_t)$, i.e. the conditional independence between actions and language instructions, given thoughts.

3.2 Hybrid Thinking

Thinking and hierarchical VLAs have demonstrated improved performance over a standard VLA [9, 10, 8]. However, generating thoughts comes at a high inference cost, as they generally consist of significantly more tokens than their action counterpart. This can significantly slow down the agent's action execution in the environment.

We hypothesize that the primary benefits of these models arise not from the generated thoughts themselves, but from the knowledge learned by the model through thought prediction. This suggests that the model refines its intuitive capabilities by internalizing the patterns present in the thoughts, akin to the development of intuitive expertise [12]. Under this hypothesis, after a learning process that is similar to those of thinking and hierarchical models, a VLA should be able to predict actions with higher accuracy, with or without the thoughts as an input.

To address the need for agents capable of producing multiple probability distributions within a single model, we introduce a new thinking strategy, *Hybrid Thinking* (HyT), designed to integrate structured reasoning with flexible policy learning.

Definition 3.1 (Hybrid Thinking) *Given a task description ℓ and the current environment observation x_t, the conditional distribution over actions a_t can be expressed as:*

$$p(a_t|x_t, \ell) = \sum_i \sum_j p(a_t, \tau^i, m^j | x_t, \ell) = \sum_i \sum_j p(a_t, \tau^i | x_t, \ell, m^j) p(m^j), \tag{1}$$

by marginalizing out thoughts τ and a "modality" variable m.

The hybrid thinking formulation enables to describe a VLA model that learns different thoughts and conditional action distributions depending on a modality variable.

3.3 Training VLAs with Hybrid Thinking

Leveraging the insights from other VLA models, we can use the hybrid thinking framework to conditionally learn three distributions:

$$p(a_t|x_t, \ell) = \underbrace{p(a_t|x_t, \ell, m^a)p(m^a)}_{\text{Act}} + \underbrace{p(a_t|x_t, \ell, \tau_t)p(\tau_t|x_t, \ell, m^\tau)p(m^\tau)}_{\text{Think}} + \underbrace{p(a_t|x_t, \tau_t, m^f)p(m^f)}_{\text{Follow}}. \tag{2}$$

The utility of each distribution is defined as follows:

- **"Act" action distribution**: following from $p(\tau = \emptyset | m^a) = 1$, is defined to predict only actions. This modality enables fast inference time, by predicting no thoughts.
- **"Think" joint distribution**: predicts thoughts and actions. This modality fosters the model to learn the thoughts distribution and to condition action predictions both on the task and the thoughts.
- **"Follow" action distribution**: follows from $p(a_t|x_t, \tau_t, m^f) = p(a_t|x_t, \ell, \tau_t, m^f)$ and $p(\tau = \emptyset | m^f) = 1$. This modality assumes that thoughts are provided to the policy and it encourages the model to follow them closely for action prediction, as they replace the language task in the inputs.

We can then assign the probabilities for different realizations of m to a set of values, which allows us to define a stable VLA formulation:

$$p(a_t|x_t, \ell) = w_\alpha \cdot p_\theta(a_t|x_t, \ell, m^a) + w_\tau \cdot p_\theta(a_t|x_t, \ell, \tau_t) p(\tau_t|x_t, \ell, m^\tau) + w_f \cdot p_\theta(a_t|x_t, \tau_t, m^f), \tag{3}$$

where $w_a = p(m^a)$, $w_f = p(m^f)$ and $w_\tau = p(m^\tau)$ are the probabilities of a modality variable following a categorical distribution and θ denotes the parameters of the model.

In order to train a HyT VLA, we define the weights w_a, w_τ, w_f as the probabilities of sampling the corresponding inputs and outputs for the model during training. This way, at each batch sampled during training, the model receives modality tokens, actions and thoughts, with probabilities that are defined by the coefficients. The model inputs and outputs are formed as in Equation 3. Then, the model's parameters are trained to minimize the cross-entropy loss on future tokens predicted autoregressively with causal mask attention [26].

During inference, we can prompt the model with different modality sets of tokens. These are defined as $m^a =$ <act>, $m^t =$ <think> and $m^f = \emptyset$[1]). Depending on the modality token received, the model is able to provide the corresponding modality's outputs, as also depicted in Figure 1.

Intuitive explanation of the "follow" distribution. Ideally, the model should be able to follow the information in the thoughts by learning the joint distribution from the "Think" modality. However, as the model is concurrently trained to predict actions, independently of the thoughts, the model could tend to ignore the thoughts. Humans may want to possibly manipulate the agent's thoughts for correcting its action predictions. Thus, it is important that changes in the thoughts have a noticeable impact on the action prediction, and the "Follow" distribution encourages such tendency.

4 Experiments

We evaluate the proposed HyT VLA in a series of simulated experiments, which we use to reliably assess the model's capabilities, and on a set of real-world tasks, which demonstrates the practical applicability of the approach.

General setup

Across all models, the task description is provided in the same format, i.e. *"What should the robot do to {task}?"*. Image inputs are 224×224 RGB images from a camera pointing at the robot's workspace. The action space is 7-dimensional and defined as: $[\Delta\mathbf{x}, \Delta\boldsymbol{\phi}, \text{gripper}]$. The end-effector position $\mathbf{x}$ and orientation $\boldsymbol{\phi}$ are controlled in delta space, while the gripper pose is in absolute value.

For all the thoughts, we adopt the same format, which includes the current subtask and, for the *moving* subtask, the main direction of the movement. This simple definition has proven effective across different models in early stages of our analysis and also in related work [9]. More elaborate thoughts could include object bounding boxes or justification of the agent's actions [8], but we decided against this as it would also significantly increase the amount of resources needed for training and the inference time for thinking models due to the increasing number of generated tokens.

For HyT, during training, the different modality tokens are sampled with equal probabilities, i.e. $p = 0.33$. During inference, we use the denomination "Fast mode" to refer to the "Act" modality during inference and "Slow mode" to refer to the "Think" modality. The mode is set by forcing the first tokens of the agent's output to be the set of tokens corresponding to $m^a =< act >$ or $m^\tau =< think >$. There is no switch between the two modalities within the same episode: we evaluate them differently, to test the different agent capabilities. Designing an orchestration mechanism that allows switching the two modalities for adaptive reasoning within the same episode is left for future work.

The training for all experiments and approaches is done using PyTorch DDP [30] on 4 A100 GPUs. Inference requires less than 20GB VRAM and is performed on a multi-instance A100 for simulated environments and on an RTX A5000 for real-world experiments.

4.1 Simulated Experiments

For the simulated experiments, we employ the ClevrSkills benchmark [31], which is based on the ManiSkill2 manipulation environments [32]. This environment includes an oracle solver to collect

[1]For the follow modality defined by m^f there's no modality tokens, but the model can still discriminate it from the other modalities as the follow modality enforces no task description.

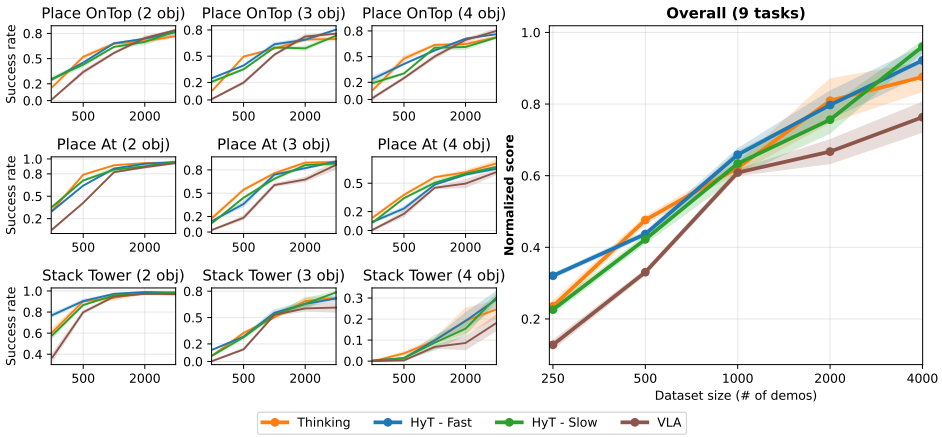


Figure 2: **Performance scalability.** On the left, success rate per task for three task groups (one per row). For each task group, agents are trained on a dataset containing a number of demonstrations from that task group that is showed on the x-axis (which uses a log scale). Then, the agent is evaluated on the three tasks of the group, averaging performance over 100 episodes $\times$ 3 checkpoints. On the right, normalized scores averaged across all the tasks and evaluations.

demonstrations, and reasoning traces from the oracle, which we can use to discriminate subtasks and extract thoughts. ClevrSkills also adopts a vacuum gripper and mostly uses simple shapes and objects, avoiding the challenges of manipulating complex objects, which may require more complex action prediction heads [33, 34].

Using the oracle, we collect a diverse set of training data which spans three task groups: *Place At*, evaluating the agent's spatial understanding, *Place OnTop*, evaluating the agent's understanding of interactions between objects, and *Stack Tower*, evaluting longer-horizon capabilities. Further details about the tasks and the datasets' definition are provided in Appendix.

For all approaches, we start from the PaliGemma-2 VLM model with 3B parameters [2], which is based on the Gemma-2 LLM [28] and on the SigLIP vision encoder [35]. We perform full-finetuning of the model with a batch size of 8 and a learning rate of $2e-5$, using the Adam optimizer [36]. For the hierarchical VLA approaches, we train two distinct PaliGemma-2 models.

During evaluation, we run the agent in the environment for 100 evaluation episodes. At inference time, we found that non-thinking VLA output actions at around $\sim$ 3Hz, while running 4 models in parallel on A100 GPUs. Thinking models are 2.5-3$\times$ slower. Hierarchical models are $\sim 4\times$ slower.

Data-scaling experiments

The ability to scale with data and compute is a crucial criterion when aiming to build general purpose agents [37]. With this experiment, we aim to verify the hypothesis that HyT VLA can develop intuitive thinking through the hybrid training strategy. If the hypothesis is correct, we expect: i) that HyT VLA in Slow mode performs as well as thinking or hierarchical VLA models, ii) that HyT VLA in Fast mode performance is better than a standard VLA.

In this section, we also verify scaling properties of HyT. We adopt a set of three datasets, one for each task group, and train models at different data scales, up to 4000 demos per task. We show results in Figure 2. The normalized scores on the right are obtained by dividing each task's success rates by the highest success rate we obtained in all experiments, such that the highest performance obtained by at least one of the approaches corresponds to 1 for all tasks (see Appendix for details). Normalizing scores this way is a common strategy when comparing agents evaluated on multiple tasks that have different performance scales [38, 39, 40].

Overall, we observe that the performance of all approaches scales similarly over time. However, Thinking and HyT models achieve higher performance earlier and maintain higher success rate over time, compared to standard VLAs, which are between 5 and 20% less performant at any data scale.

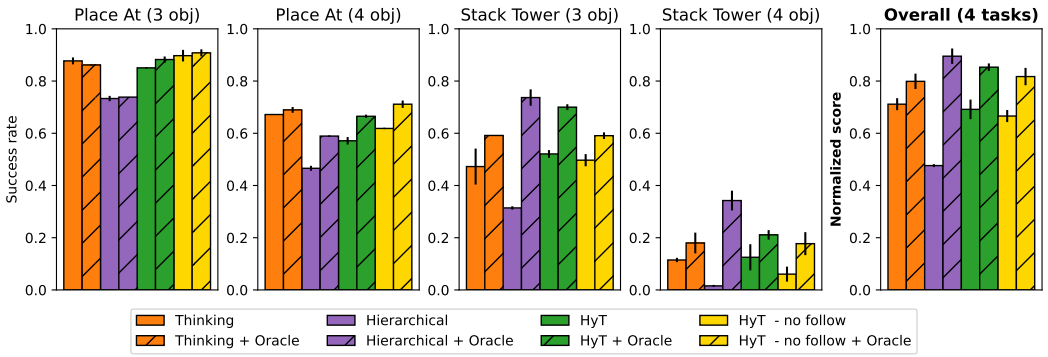


Figure 3: **Intervening on thoughts.** Comparing performance across thinking VLA, when using their own generated thoughts (default) or using thoughts generated by an oracle. Oracle thoughts are only provided for certain parts of tasks, i.e. the "move to X" subtasks. Actions are predicted by the VLA.

The fact that HyT - Fast is performing as well as Thinking and HyT - Slow VLAs confirm our hypothesis that directly predicting actions in Fast mode benefits from the hybrid thinking training.

Scale matters! As a matter of fact, the performance increases with larger datasets across all tasks. The models saturate performance (close to 100%) on the easier tasks and start solving the most complex task, Stack Tower with 4 objects, at larger data scales.

Intervenability experiments

One useful feature of Thinking and Hierarchical models is the possibility to intervene on the model's thoughts. One way of intervening is to replace the agent's generated thoughts with human-determined thoughts to correct or steer the agent's behavior towards the solution of a task. Ideally, HyT models should also be intervenable. In particular, the addition of the "Follow" modality in Equation 3 should encourage the agents to condition its actions on the thoughts.

To verify this empirically, we train a set of models on a multitask dataset of 3000 demos.[2] At this data scale, the models should be able to easily solve easier tasks and have modest success on harder tasks. In order to improve success on harder tasks, we replace the agent's thoughts with a set of "Oracle thoughts", which we extract using the code from the the ClevrSkills' oracle. Oracle thoughts replace the agent's thoughts only during moving subtasks, i.e. "move to location X". This is because the oracle has precise conditions for picking and placing. A learned policy not always satisfies them, while still being successful, causing the agent to get stuck.

We present results in Figure 3 for four of the most complex tasks of our benchmark. We also compare to an ablation of HyT that does not include the "Follow" component in Eq. 3, to verify whether the agent benefits or not from this component during training. We observe that overall all methods improve when provided oracle thoughts. In particular, hierarchical VLAs, which generally tend to perform worse than the other models, significantly improve in Stack tasks thanks to the oracle thoughts. Thinking and HyT VLAs receives similar improvements from using oracle thoughts, proving that HyT is definitely intervenable. HyT ablation performs slightly worse than its counterpart, showing a minor but still beneficial increase in performance when including the "Follow" modality during training, both with and without the oracle thoughts.

4.2 Real-world Experiments

One of the major benefits of the HyT approach is its capability to retain higher performance at a lower inference time, which is the same as with standard VLAs. This benefit mainly makes sense in real-world use-cases, where faster execution is important to carry out tasks quickly and to increase

[2]The choice of using a multitask dataset here is mostly to save compute. Hierarchical models, for instance, require training two models for each dataset. This is also the reason why we omitted them from the scaling experiments.

Table 1: Success rates on the real-world tasks and number of successful trials. Some pictures of the agent performing the tasks are available in Figure 1. Additional details are provided in Appendix.

	FT-OpenVLA	**HyT- Fast**
In-distribution tasks	**52**% (13/25)	**72**% (18/25)
Place the banana in the green bowl	70% (7/10)	70% (7/10)
Place the red cube in the brown bag	50% (3/6)	100% (6/6)
Place the tomato left of the lettuce	60% (3/5)	60% (3/5)
Place the zucchini in front of the green cube	0% (0/4)	50% (2/4)
Out-of-distribution tasks	**29**% (7/24)	**54**% (13/24)
Place the rubber duck in the green bowl	40% (4/10)	20% (2/10)
Place the mushroom in the brown bag	0% (0/6)	100% (6/6)
Place the purple die left of the lettuce	0% (0/4)	50% (2/4)
Place the zucchini in front of the red hexagon	75% (3/4)	75% (3/4)
Overall	**41**% (20/49)	**63**% (31/49)

the perceived quality of the agent's execution. Thus, it is important to verify whether the findings in simulation about HyT transfer to real-world scenarios.

For our real-world experiments, we collected a dataset comprising 320 trajectories using a robotic setup featuring an UFactory xArm 6 with a flexible two-fingered gripper, operating on a white tabletop. The agent observes the environment through RGB images captured by a RealSense D435 camera, positioned at a corner of the table. For the models, we start from the pre-trained OpenVLA model with 7B parameters [4] and compare directly to it. We perform LoRA fine-tuning with rank 32 [41], a batch size of 8 and a learning rate of $5e-4$, using the Adam optimizer [36].

Our evaluation spans two categories of tasks: in-distribution and out-of-distribution. The in-distribution set includes tasks for which the dataset contains at least 10 demonstrations. For the out-of-distribution set, we modify the in-distribution tasks by altering certain elements, such as the object to be picked or the reference object for placement, ensuring the agent encounters novel scenarios not present in the training data. See the Appendix for additional details.

The robot control frequency is ~ 3 Hz with no thinking and ~ 1 Hz with thinking. Thus, due to the long times required for thinking models evaluation, we study only the OpenVLA model fine-tuned on our dataset (using the original code) and HyT in Fast mode. The results, shown in Table 1, show that HyT overall significantly outperforms OpenVLA, especially in out-of-distribution tasks.

From a qualitative perspective, we notice that OpenVLA and HyT have similar flaws, e.g., they tend to pick objects with the wrong orientation. However, HyT tends to be more precise when reaching picking and placing positions, e.g. it never reached for the wrong object while OpenVLA did, eventually leading to a noticeable performance gap.

5 Conclusion

Thinking strategies for VLAs [8, 9] have shown important benefits in terms of performance and intervenability over standard VLAs, with the drawback of slower inference. In this work, we support the idea that the thinking process can be "internalized" by the model, developing some form of expert intuition [12]. We proposed the Hybrid Thinking framework that enables the agent to use different training and inference modalities. This enables the possibility of learning from thoughts, while also being able to predict actions directly. As empirically validated through simulated and real-world experiments, Hybrid Thinking VLAs can obtain data-efficient performance and intervenability, along with fast inference.

Limitations. Thoughts in robotics require an user to design a thought structure that is beneficial to the agent. Automated reasoning approaches to obtain useful thoughts may be explored [18]. In this work, we adopted large 3B and 7B parameters models. As we deploy faster models for computing actions on edge devices, we should aim to achieve similar performance with compute-efficient models [42].

References

[1] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024. URL https://arxiv.org/abs/2310.08864.

[2] A. Steiner, A. S. Pinto, M. Tschannen, D. Keysers, X. Wang, Y. Bitton, A. Gritsenko, M. Minderer, A. Sherbondy, S. Long, S. Qin, R. Ingle, E. Bugliarello, S. Kazemzadeh, T. Mesnard, I. Alabdulmohsin, L. Beyer, and X. Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL https://arxiv.org/abs/2412.03555.

[3] S. Tong, E. Brown, P. Wu, S. Woo, M. Middepogu, S. C. Akula, J. Yang, S. Yang, A. Iyer, X. Pan, Z. Wang, R. Fergus, Y. LeCun, and S. Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms, 2024. URL https://arxiv.org/abs/2406.16860.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[8] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[9] L. X. Shi, B. Ichter, M. Equi, L. Ke, K. Pertsch, Q. Vuong, J. Tanner, A. Walling, H. Wang, N. Fusai, A. Li-Bell, D. Driess, L. Groom, S. Levine, and C. Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

[10] NVIDIA, :, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

[11] D. Kahneman. *Thinking, fast and slow*. macmillan, 2011.

[12] D. Kahneman and G. Klein. Conditions for intuitive expertise: a failure to disagree. *Am Psychol*, 64(6): 515–526, Sept. 2009.

[13] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[14] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng, F. Feng, and J. Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation, 2024. URL https://arxiv.org/abs/2409.12514.

[15] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

[16] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts, 2023. URL https://arxiv.org/abs/2210.03094.

[17] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[18] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[19] A. Havrilla, Y. Du, S. C. Raparthy, C. Nalmpantis, J. Dwivedi-Yu, M. Zhuravinskyi, E. Hambro, S. Sukhbaatar, and R. Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

[20] H. Shao, S. Qian, H. Xiao, G. Song, Z. Zong, L. Wang, Y. Liu, and H. Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning, 2024. URL https://arxiv.org/abs/2403.16999.

[21] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, A. Handa, M.-Y. Liu, D. Xiang, G. Wetzstein, and T.-Y. Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2503.22020.

[22] Y. Deng, Y. Choi, and S. Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step, 2024. URL https://arxiv.org/abs/2405.14838.

[23] P. Yu, J. Xu, J. Weston, and I. Kulikov. Distilling system 2 into system 1, 2024. URL https://arxiv.org/abs/2407.06023.

[24] D. Su, S. Sukhbaatar, M. Rabbat, Y. Tian, and Q. Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces, 2025. URL https://arxiv.org/abs/2410.09918.

[25] M. Zhu, Y. Zhu, J. Li, J. Wen, Z. Xu, Z. Che, C. Shen, Y. Peng, D. Liu, F. Feng, and J. Tang. Language-conditioned robotic manipulation with fast and slow thinking, 2024. URL https://arxiv.org/abs/2401.04181.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

[27] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[28] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, J. Ferret, P. Liu, P. Tafti, A. Friesen, M. Casbon, S. Ramos, R. Kumar, C. L. Lan, S. Jerome, A. Tsitsulin, N. Vieillard, P. Stanczyk, S. Girgin, N. Momchev, M. Hoffman, S. Thakoor, J.-B. Grill, B. Neyshabur, O. Bachem, A. Walton, A. Severyn, A. Parrish, A. Ahmad, A. Hutchison, A. Abdagic, A. Carl, A. Shen, A. Brock, A. Coenen, A. Laforge, A. Paterson, B. Bastian, B. Piot, B. Wu, B. Royal, C. Chen, C. Kumar, C. Perry, C. Welty, C. A. Choquette-Choo, D. Sinopalnikov, D. Weinberger, D. Vijaykumar, D. Rogozińska, D. Herbison, E. Bandy, E. Wang, E. Noland, E. Moreira, E. Senter, E. Eltyshev, F. Visin, G. Rasskin, G. Wei, G. Cameron, G. Martins, H. Hashemi, H. Klimczak-Plucińska, H. Batra, H. Dhand, I. Nardini, J. Mein, J. Zhou, J. Svensson, J. Stanway, J. Chan, J. P. Zhou, J. Carrasqueira, J. Iljazi, J. Becker, J. Fernandez, J. van Amersfoort, J. Gordon, J. Lipschultz, J. Newlan, J. yeong Ji, K. Mohamed, K. Badola, K. Black, K. Millican, K. McDonell, K. Nguyen, K. Sodhia, K. Greene, L. L. Sjoesund, L. Usui, L. Sifre, L. Heuermann, L. Lago, L. McNealus, L. B. Soares, L. Kilpatrick, L. Dixon, L. Martins, M. Reid, M. Singh, M. Iverson, M. Görner, M. Velloso, M. Wirth, M. Davidow, M. Miller, M. Rahtz, M. Watson, M. Risdal, M. Kazemi, M. Moynihan, M. Zhang, M. Kahng, M. Park, M. Rahman, M. Khatwani, N. Dao, N. Bardoliwalla, N. Devanathan, N. Dumai, N. Chauhan, O. Wahltinez, P. Botarda, P. Barnes, P. Barham, P. Michel, P. Jin, P. Georgiev, P. Culliton, P. Kuppala, R. Comanescu, R. Merhej, R. Jana, R. A. Rokni, R. Agarwal, R. Mullins, S. Saadat, S. M. Carthy, S. Cogan, S. Perrin, S. M. R. Arnold, S. Krause, S. Dai, S. Garg, S. Sheth, S. Ronstrom, S. Chan, T. Jordan, T. Yu, T. Eccles, T. Hennigan, T. Kocisky, T. Doshi, V. Jain, V. Yadav, V. Meshram, V. Dharmadhikari, W. Barkley, W. Wei, W. Ye, W. Han, W. Kwon, X. Xu, Z. Shen, Z. Gong, Z. Wei, V. Cotruta, P. Kirk, A. Rao, M. Giang, L. Peran, T. Warkentin, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, D. Sculley, J. Banks, A. Dragan, S. Petrov, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, S. Borgeaud, N. Fiedel, A. Joulin, K. Kenealy, R. Dadashi, and A. Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[31] S. Haresh, D. Dijkman, A. Bhattacharyya, and R. Memisevic. Clevrskills: Compositional language and visual reasoning in robotics, 2024. URL https://arxiv.org/abs/2411.09052.

[32] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills, 2023. URL https://arxiv.org/abs/2302.04659.

[33] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

[34] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[35] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

[36] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[37] R. Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1):38, 2019.

[38] J. Fan. A review for deep reinforcement learning in atari:benchmarks, challenges, and solutions, 2023. URL https://arxiv.org/abs/2112.04145.

[39] P. Mazzaglia, T. Verbelen, B. Dhoedt, A. Courville, and S. Rajeswar. Genrl: Multimodal-foundation world models for generalization in embodied agents, 2024. URL https://arxiv.org/abs/2406.18043.

[40] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024. URL https://arxiv.org/abs/2310.16828.

[41] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[42] T. Dao and A. Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

[43] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

A Appendix

A.1 Details about Figure 1

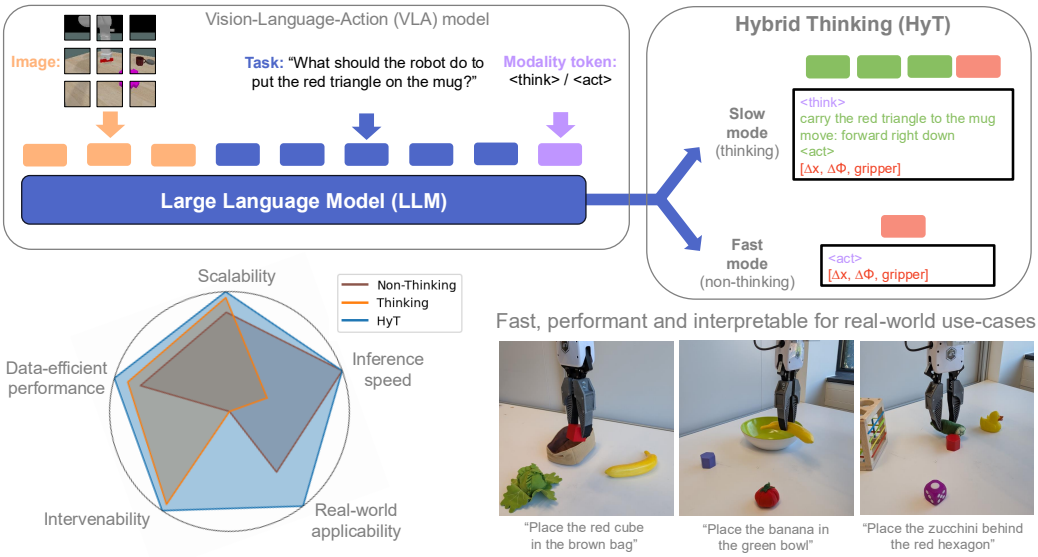


Figure 4: **Hybrid Thinking VLAs.** The hybrid thinking framework enables vision-language-action models to switch between fast execution and thoughtful - slow - execution. In addition to an image of the scene and a language description of the task, the agent receives a modality token, which conditions the output of the model. We observed that, even in the fast mode (high inference speed), the model has high performance, both in low-data (data-efficiency) and large-data (scalability) regimes. The high performance and fast inference speed enable fast and performant deployment in real-world scenarios (real-world applicability). The improved performance obtained when using 'oracle' thoughts together with the slow mode of HyT demonstrate the agent's interpretability and the possibility to intervene on the thoughts for behavior's correction (intervenability).

In the caption of Figure 4, we provide a detailed explanation of the HyT framework and performance. The radar plot in the Figure illustrates the results of standard VLAs as 'Non-thinking', thinking VLA as 'Thinking', and HyT. The data used for the radar plot is presented in Table 2 and obtained as follows:

- **Inference speed**: measured on (fractions of) A100 GPUs, while performing inference on the ClevrSkills tasks. HyT is in Fast mode;
- **Scalability**: overall normalized scores after training on the 4000 trajectories datasets on ClevrSkills (as in Figure 2). HyT results are obtained in Fast mode;
- **Data-efficient**: overall normalized scores after training on the 1500 trajectories multitask dataset and evaluating on ClevrSkills (see Table **??**). HyT results are obtained in Fast mode;
- **Intervenability**: overall normalized scores when intervening on the thoughts (see Fig 3). HyT results must be obtained in Slow mode. Non-thinking is 'NA' as it doesn't allow interventions on the thoughts;
- **Real-world**: overall success rate in real-world tasks (see Table 1). HyT results are obtained in Fast mode. Thinking VLAs are excluded from the comparison due to time constraints (the inference process is 3× slower).

Table 2: Max scores for different systems across evaluation dimensions

	Inference speed	**Scalability**	**Data-efficient**	**Intervenability**	**Real-world**
Non-thinking	3 Hz	0.76	0.33	NA	0.41
Thinking	1 Hz	0.88	0.37	0.80	NA
HyT	3 Hz	0.92	0.42	0.85	0.64

A.2 Simulation experiments

Tasks definition The tasks are defined as follows:

- *Place At* tasks evaluate the agent's spatial understanding. The tasks require the agent to bring an object to a location specified in relation to another object location, i.e. left/right/behind/in front of the object. The objects used are mainly simple-shaped objects of different colors as in [16].
- *Place OnTop* tasks evaluate the agent's understanding of object interactions, as they require placing an object on top of another in a stable position. The set of objects used contains simple shapes, but also more complex objects from YCB, such as mugs, wooden blocks, bowls, etc.
- *Stack Tower* evaluates the long-horizon capabilities of the agent, as it requires multiple "place on top" with simple object shapes, where the order of the objects in the stack is defined in the prompt.

For each task, we train and test the agent with three variants, containing varying number of objects. In *Place OnTop* and *Place At* the number of additional objects is only distracting or increasing the amount of clutter in the scene, e.g. making placing objects on the table harder. For the *Stack Tower* tasks more objects also require more actions, as the agent should stack all the objects present in the scene.

Evaluation details At each episode, objects and positions are resampled randomly (but consistently among evaluation runs), making no evaluation environment exactly the same as any of the training examples. The agent is, thus, required to generalize actions to new settings in order to succeed. For each model, we evaluate 3 checkpoints taken at epochs 5,7 and 10 during training. In general, we found no strong correlation between validation losses and performance on the task. Though, with additional training after 10 epochs, models tend to start overfitting.

Dataset definition. For each task group, we collected a set of 4000 demo trajectories composed as follows: 1000 demos in the 2 objects task, 2000 demos in the 3 objects task, 1000 demos in the 4 objects task. Then, to train agents with different sizes of the dataset, we subsample fixed subsets of trajectories from each dataset. For the multitask dataset, the task group datasets are subsampled and then aggregated.

Example 1: 'Stack Tower' task group dataset with 1000 demos (scaling experiments). The 4000 demos dataset is subsampled, maintaining an equal ratio of trajectories per task. The number of trajectories per task will be: 250 demos in the Stack Tower with 2 objects task, 500 demos in the Stack Tower with 3 objects task, 250 demos in the Stack Tower with 4 objects task.

Example 2: multitask dataset with 3000 demos (intervenability experiments). For each task group, we subsample a dataset of 1000 demos (see Example 1). Then, 3000 demos are obtained by aggregating all the task groups' demos.

Extracting thoughts. In ClevrSkills' demonstrations [31] a variety of solvers is applied to the task, following a pre-specificied order - the oracle's plan. Each solver takes executes one subtask from the overall plan and the solver parameters can be recovered in the demonstrations. In order to create thoughts, we can use the ClevrSkills library to transform each subtask from the solver into a natural language instruction, e.g. "Move to X" or "Pick up Y". Then, for moving instructions - e.g. "Move to..." and "Carry Z to..." - we extract the motion direction of the agent. Similarly to [9], this is obtained at each timestep by computing the distance between the current end-effector position and the position at the end of the motion subtask. This is then transformed into language, in the form of "left/right", "forward/backward" and "up/down" instructions, as in [8]. Finally, for distances that are less than 1 cm the agent receives a "close" moving instruction. Two examples of thoughts and actions, in the HyT format, are provided in Figure 5.

Gripper information. The ClevrSkills benchmark adopts a vacuum-gripper robot which creates a vacuum in the gripper's suction cups to maintain contact with objects. Compared to a fingers-based gripper, a suction gripper's state, i.e. open or closed, cannot be easily assessed through images. Thus, in order for the agent to be able to perform grasping actions reliably, we concatenate the state of the gripper in language form to the task description (or to the thoughts for the hierarchical low-level VLA). The gripper state information we pass to the agent is: (i) whether the gripper suction cups are making contact with something, (ii) whether the gripper was on, grasping something, or not. There's no need for the gripper information in the real-world experiments, as we use a fingers-based gripper, where the gripper state is visually observable.

(a) (b)

Figure 5: Examples of thoughts and actions for a moving (a) and a non-moving (b) subtasks, expressed in language format following the format used for HyT.

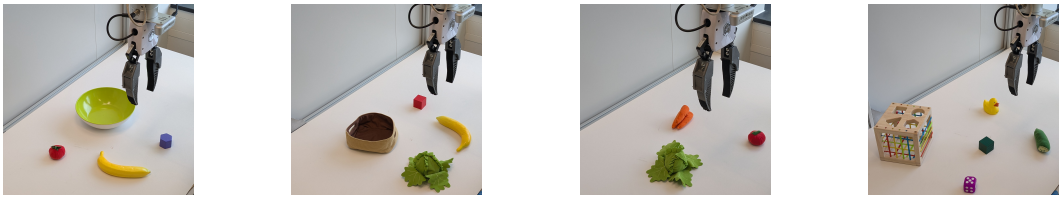
A.3 Normalizing scores

In Figures 1 and 3 we used 'normalized scores' to provide aggregate results across different tasks. This is typically done when different tasks have results in different scales across a benchmark [40, 39, 38]. In Table 3, we report the highest success rates obtained by any approach in each task. We see that for some tasks the success rate is significantly lower than for others, e.g. Stack Tower with 3 and 4 objects and Place At with 4 objects are harder to solve. Thus, when showing 'overall' performance, we rescale the results on each task in [0,1] by dividing them by the maximum obtained in that task - that is shown in the Table. Then, results are aggregated into 'normalized scores' providing a more insightful view on each approach's relative performance [43].

Table 3: Highest success rate obtained by any approach (at any scale) and used to normalize scores.

Category	2 Objects	3 Objects	4 Objects
Place At	0.96	0.91	0.71
Place OnTop	0.84	0.80	0.80
Stack Tower	0.99	0.79	0.34

A.4 Real-world experiments



(a) (b) (c) (d)

Figure 6: The setup for some of the real-world tasks: (a) banana in green bowl (b) red cube in brown bag (c) zucchini in front of green cube (d) tomato left of lettuce.

Dataset preparation. The dataset for real-world experiments is collected via human tele-operation, using a VR set and a controller to map the human motions and intentions, e.g., closing the gripper, to the robot. After collecting the trajectories, we apply some basic pre-processing operations. First, we take the sequences collected at 60 Hz, and we subsample them at 10 Hz. Then, we extract actions in the format $[\Delta x, \Delta\phi, \text{gripper}]$ from the end-effector and gripper positions of the demonstrations. Finally, we filter out no-motion operations, i.e. having Δx and $\Delta\theta$ equal to zero.

Extracting thoughts. For extracting the subtasks to use in the thoughts, we use a simple heuristic-based approach. This is based on the assumption that all the tasks in the dataset are in the format 'place the {obj1} {position} the {obj2}', e.g. 'place the {zucchini} {in front of} the {green cube}'. We identify the position when the robot closes the gripper as the *grasping position*, and the position when the robot opens the gripper as the *releasing position*. Then, we identify the following 3 keyframe moments: (i) the moment when the agent reaches within 3 cms distance from the grasping position,

(ii) the grasping moment, (iii) the moment when the agent reaches within 3 cms distance from the releasing position. Finally, the subtasks are defined in the trajectory as follows:

- *Move to the {obj1}*: between the start of the trajectory and keyframe (i);
- *Pick up the {obj1}*: between keyframe (i) and keyframe (ii);
- *Move {position} the {obj2}*: between keyframe (ii) and keyframe (iii);
- *Place object {position} the {obj2}*: between keyframe (iii) and the end of the trajectory.

Also, to have the same thought structure as in the ClevrSkills' experiments, we concatenate a 'move: {direction}' to the moving subtasks, where the direction is computed using the end-effector position (see Section A.2).

Evaluation. During evaluation, we limit the execution time to 150 agent steps, which is $\sim$ 2 minutes per trajectory, considering 3 actions/second from the model and a control loop running at $\sim$ 2Hz for stable motions. In case of failed grasps, we allow up to 3 attempts to the agent, before assessing the outcome of the episode. In the following, you find a detailed description of how each in-distribution task is executed and randomized. For out-of-distribution tasks, we follow the same procedures, but we swap one of the main actors in the scene (e.g. the object to pick or the placing target). See Figure 6) for reference. Note: fruits are stuffed toys, while vegetables are hard foam models.

- *Place the banana in the green bowl.* The scene includes a green bowl and three objects: a banana, a tomato, and a blue hexagon, each with predefined positions. We conduct five trials with the banana starting in one location, and five more from a different location. In each location, the banana appears in three orientations: vertical (2/5 trials), horizontal (2/5 trials), and diagonal (1/5 trial).
- *Place the red cube in the brown bag.* The setup features a brown bag and three objects: a red cube, a banana, and a lettuce leaf, all placed at specific locations. We run two trials for each location of the red cube. Its orientation varies between being aligned with the robot's base and being tilted.
- *Place the tomato left of the lettuce.* The scene contains three objects: a lettuce leaf, a carrot, and a tomato. Their positions are randomized within defined regions, though the overall layout remains consistent. We perform five trials with these randomized placements. A trial is only considered successful if the tomato ends up on the table, near the lettuce, and to its left.
- *Place the zucchini in front of the green cube.* The environment includes five objects: a shape sorting box, a zucchini, a purple die, a rubber duck, and a green cube. Object positions are randomized within certain bounds, while maintaining the general layout. We carry out four trials with these randomized setups. Success is defined as the zucchini being placed on the table, close to and in front of the green cube.